Towards Radical Agent-Oriented Software Engineering Processes Based on AOR Modeling

Gerd Wagner
Eindhoven University of Technology,
Faculty of Technology Management
G.Wagner@tm.tue.nl

Kuldar Taveter
VTT Information Technology
(Technical Research Centre of Finland)
Kuldar.Taveter@vtt.fi

Abstract

We propose a new agent-oriented software engineering process, called RAP, which follows the Rational Unified Process (RUP) in many ways, but is based on Agent-Object-Relationship (AOR) modeling instead of object-oriented modeling. Two particular features of the proposed methodology are: it is supported by a foundational ontology, and it employs a certain form of agent-based discrete event simulation for achieving more agility in the development process.

1. Introduction

A *Radical Agent-Oriented Process (RAP)* defines a software engineering process[1] based on agent-oriented modeling of both the system to be engineered and the engineering process itself using the Agent-Object-Relationship (AOR) modeling language proposed in [1]. In AOR modeling, the agents in a problem domain are distinguished from the (non-agentive) objects. The agents' actions, perceptions, commitments and claims, as well as their rights and duties, are explicitly included in the model.

The RAP/AOR methodology is based on the Business Agents' approach proposed in [2] and can be viewed as an agent-oriented refinement of the *Rational Unified Process (RUP)* [3]. It aims at achieving more *agility* than the RUP by using simulation for early testing of analysis and design models, and by adopting an agent-oriented project management approach.

Agile methodologies have received much attention recently (see [4]). They emphasize the value of lightweight ad-hoc processes based on rapid prototyping and de-emphasize the value of (detailed) modeling on which they blame the heavy weight and inflexibility of traditional methodologies and the RUP. While we acknowledge the significance of agility, we disagree with their analysis that blames modeling as the source of inflexibility. Rather, we agree with the *Model-Driven Architecture (MDA)* approach of the OMG [5] where modeling is identified as the core of state-of-the-art software engineering that is scientifically well-founded. When a model-driven approach includes early testing of models by means of simulation, agility is achieved even without setting a focus on code and rapid prototyping.

Unlike many other agent-oriented methodologies, RAP/AOR is more concerned with distributed information systems (such as enterprise resource planning and supply chain management systems) and not so much with Artificial Intelligence systems. This difference implies that we are not so ambitious about capturing human-like intelligence features such as *desires* and *intentions*, or sophisticated forms of pro-active behavior. Rather, in RAP/AOR we focus on declarative models of communication and interaction founded on reactive behavior and on the basic mental state components of *beliefs*, *perceptions* and *commitments*.

2. Ontological foundations

The ontological foundation of the RAP/AOR concepts is provided by the Unified Foundational Ontology (UFO) proposed in [6]. In addition to a foundation layer, called UFO-A, and the perdurant ontology layer UFO-B, UFO includes an agent ontology layer, UFO-C, which is the basis of AORML. While beliefs and perceptions are categorized as *mental moments* (endurants that existentially depend on one agent, their 'bearer'), commitments are categorized as *social moments* (endurants that existentially depend on several agents).

Notice that the UML concept of an *actor* corresponds to the UFO concept of an *agent role type*. For instance, the actor type Employee is a role subtype of the base type Person. In many cases, an *actor type* corresponds to the UFO concept of an *agent role mixin type*. For instance, the actor type BookingClerk can be partitioned into HumanBookingClerk (being a role subclass of Person) and SoftwareAgentBookingClerk (being a role subclass of SoftwareAgent).

[1] Strictly speaking, the RAP defines a *process type family* whose members are *process types* that can be instantiated by different *process individuals*. It is common practice, though, to use the term 'process' ambiguously both at the level of types and at the level of instances.

3. The AOR modeling language

The AOR modeling language is based on an ontological distinction between active and passive entities, that is, between agents and (non-agentive) objects of the real world. The agent metaphor subsumes *artificial* (software and robotic), *natural* (human and animal) as well as *social/institutional* agents (groups, organizations).

In AORML, an entity is an agent, an event, an action, a claim, a commitment, or an ordinary object. Only agents can communicate, perceive, act, make commitments and satisfy claims. Objects are passive entities with no such capabilities. Besides human and artificial agents, AORML also includes the concept of *institutional agents,* which are composed of a number of other agents that act on their behalf. Organizations and organizational units are important examples of institutional agents.

Figure 1 shows the most important elements of external (i.e., modeled from the perspective of an external observer) AOR state structure modeling. There is a distinction between *action events* and *non-action events*, and between a *communicative* action event (or *message*) and a *non-communicative* action event. Figure 1 also shows that a *commitment/claim* is coupled with the action event that fulfils that commitment (or satisfies that claim).

AOR state structure modeling can be defined as a UML Profile, i.e., it is a conservative extension of UML class modeling [7].

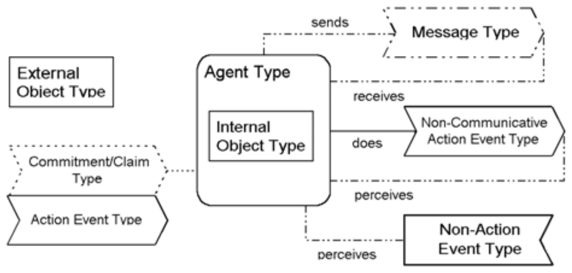

Figure 1. The core state structure modeling elements of external AOR diagrams.

The most important behavior modeling element of AORML are ***reaction rules***, which are used to express *interaction patterns*. As is shown in Figure 3, a reaction rule is visualized as a circle with incoming and outgoing arrows drawn within the agent rectangle whose reaction pattern it represents. Each reaction rule has exactly one incoming arrow with a solid arrowhead: it specifies the triggering event type. In addition, there may be ordinary incoming arrows representing state conditions (referring to corresponding instances of other entity types). There are two kinds of outgoing arrows: one for specifying mental effects (changing beliefs and/or commitments) and one for specifying the performance of actions.

4. The methodology

A RAP defines *who* is doing *what* (producing *which artifact* using *which language*), *how*, *when* and *why*. These interrogatives refer to the following modeling elements:

- *who* is doing: ***agent role types***;
- *what* is done: ***action/activity types***;
- *what* is produced: ***artifact types***;
- *how* and *when* is something done: ***behavioral patterns***;
- *why* is something done: ***purposes/goals***.

The RAP/AOR viewpoint modeling framework is based on the ideas of the Zachman framework [8] and is well-aligned with the Model-Driven-Architecture (MDA) framework of the Object Management Group [5]. It consists of a matrix with three rows representing its abstraction/modeling levels of *conceptual modeling*, *computational design*, and *implementation*, and five columns representing the viewpoint aspects *actors*, *motivation*, *interaction*, *information*, and *behavior*. Each cell in this matrix, if applicable, represents a specific viewpoint, such as *Conceptual Interaction Modeling*, *Computational Information Design*, or *Behavior Implementation*.

In the sequel, we briefly describe each type of model by using a case study of business-to-business e-commerce. The case study is based on the RosettaNet standard [9], whose "Request Quote" Partner Interface Process® (PIP) enables a buyer to request a product quote from a provider and a provider to respond with a quote. The prices and product availability reflected in a quote may be influenced by an existing or potential relationship between a buyer and provider. We now discuss the conceptual modeling of this problem domain in terms of the RAP/AOR viewpoint-modeling framework.

4.1. The organizational and informational aspects

The ***organizational aspect*** concerns the modeling of *actors (within organizations)*, i.e. agents and agent types and relationships between them. The organizational aspect is captured by *AOR agent models* which are represented by using agent diagrams. An agent model includes all agent (role) types of a business domain. The purpose of an agent model is to give an overview of the business system viewed as a multiagent system. The agent model of the domain of business-to-business e-commerce represents the agent role types Buyer and Seller with their respective internal agent types.

The ***informational aspect*** deals with the modeling of *beliefs* of the agents. The informational aspect is captured by *conceptual AOR information models*. An AOR information model describes agent, object, (action) event, and commitment types, as well as their relationships with each other. AOR information modeling is based on UML class diagrams (see [10]). In the information model of the case study, the object types PurchaseOrder, Quote, and Invoice are *shared* between agents of the types Buyer and Seller, while the object instances :SellerDatabase and :ProductDatabase are represented exclusively within agents of the types Buyer and Seller, respectively. An object of the type QuoteLineItem satisfies one of the following ***status predicates***: isBid, isNoBid, and isPending, while an object of the type ProductLineItem is characterized by the status predicate isAccept, isReject, or isPending.

4.2. The interactional aspect

The ***interactional aspect*** concerns the modeling of *interactions* and *communication* between the agents. It is captured by *conceptual AOR interaction models* which are represented by using *interaction frame diagrams* of AORML. According to [1], an ***interaction frame diagram*** of AORML provides *a static picture* of the possible interactions between two (types of) agents without modeling any specific process instance. The interaction frame diagram covering the business process types of quoting and ordering between the agent role types Buyer and Seller is depicted in Figure 2 by using the notation described in Figure 1.

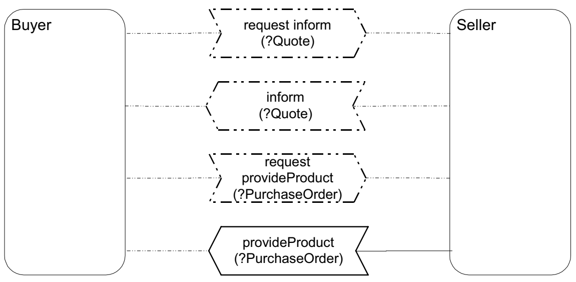
Buyer
Seller
request inform (?Quote)
inform (?Quote)
request provideProduct (?PurchaseOrder)
provideProduct (?PurchaseOrder)

Figure 2. The interaction frame between the agent roles Buyer and Seller.

4.3. The functional / motivational aspect

The ***functional*** / ***motivational aspect*** deals with the modeling of the types of *activities* performed by the agents (specifying *what* has to be done) and with the modeling of the *goals* defined for the activity types. Within this aspect, we declare for each activity type its name, optional input parameters, and an optional goal which is defined in terms of the input parameters. The functional / motivational aspect is captured by *activity diagrams* of the extended AORML [2].

An ***activity type*** (*task* in [11]), like "Confirm quote" in Figure 3, is defined as a prototypical job function in an organization that specifies a particular way of doing something [11]. It seems natural to allow specifying the start of a first-level activity in the action part of a reaction rule. For example, in Figure 3 an activity of the type "Manage quoting", which is visualized as a rectangle with rounded left and right sides, is started by reaction rule R1 in response to receiving a message containing a request for quote. As is shown in Figure 3, an activity of the type "Manage quoting" consists of sequential subactivities of the types "Process product items" and "Confirm quote".

Each activity type represented in Figure 3 is characterized by the corresponding goal that its instance is trying to achieve. For example, the goal of an activity of the type "Process product items" can be represented informally as "For each product item included by the request for quote is known whether it is to be bid or not". This goal can be formalized by means of the Object Constraint Language (OCL) of UML [10] as is shown in Figure 3 for the activity type "Process product items".

4.4. The behavioral aspect

The **behavioral aspect** of the RAP/AOR viewpoint modeling framework addresses the modeling of business behavior (specifying *when*, *how*, and *under what conditions* activities have to be performed). It also deals with the decomposition of activities into actions. The behavioral aspect is captured by the *conceptual AOR behavior models* which can be represented by AORML *activity diagrams* proposed in [2].

In order to turn activity diagrams of the functional / motivational aspect into activity diagrams of the behavioral aspect, they are elaborated on by introducing into them behavioral constructs by means of reaction rules. In [2], we have shown that AORML activity diagrams allow representing 16 out of the 19 behavioral workflow patterns defined in [12]. The behavioral model of the business process type of quoting modeled from the perspective of the agent role Seller is represented in Figure 3. As the figure shows, the function model has been complemented by the behavioral construct of the type "For-each loop". In addition, elementary actions that the activity types "Process product item" and "Confirm quote" consist of have been specified.

According to the behavioral construct of the type "For-each loop" mentioned above, upon the start of an activity of the type "Process product items", its subactivity of the type "Process product item" is performed for each instance of the object type QuoteLineItem for which the precondition quote = q evaluates to true. The precondition limits the set of QuoteLineItems for which the subactivity is performed to the ones belonging to the instance of Quote that is identified by the value of the input parameter q. When all subactivities of the type "Process product item" have ended, the enclosing activity of the type "Process product items" also ends.

The subactivity "Process product item" in Figure 3 checks the availability of the given product item that is specified by the input parameter item of the type QuoteLineItem. If the product item is available in the quantity requested, the status of the QuoteLineItem is set to isBid. In the opposite case, the status of the QuoteLineItem is changed to isNoBid. In both cases, the attributes of the QuoteLineItem are accordingly updated.

A reaction rule may have more than one triggering event: in such a case the rule is triggered only when all specified triggering events have occurred (without any required order). For example, in an activity state of the type "Confirm quote" represented in Figure 3, the triggering events of reaction rule R4 are the starting event of the current activity of the type "Confirm quote" and an approval of the quote by a human agent instance of the type Clerk.

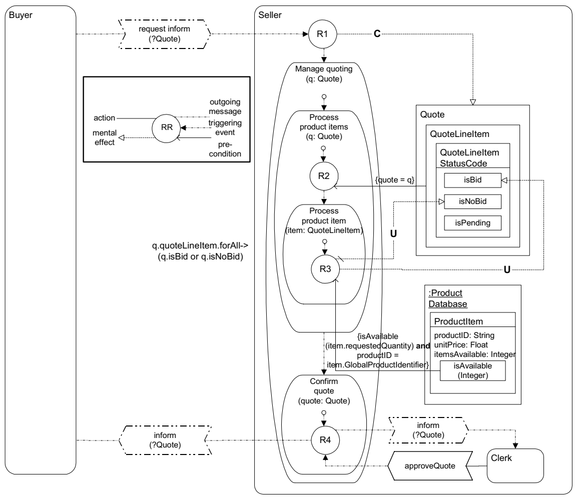

Figure 3. Behavior model of the quoting business process from the Seller perspective.

5. The role of simulation

It is shown in [2] that activity diagrams of both the functional / motivational aspect and behavioral aspect are executable. This facilitates the use of simulation in the RAP/AOR methodology. We have shown in [13] that, with some minor extensions, AOR models can be used for a certain form of agent-based discrete event simulation, called *Agent-Object-Relationship Simulation (AORS)*. In RAP/AOR, we employ AORS for achieving more agility in the software engineering process by getting feedback from the execution of models before they are implemented in a target technology platform. AORS allows animating and testing conceptual AOR behavior models, as well as AOR behavior design models An AORS system includes an environment simulator that is responsible to simulate exogenous events and the causality laws of the physical environment. Other actors of the problem domain can be simulated with various degrees of realism.

6. Conclusions

In this paper we have introduced the RAP/AOR methodology for agent-oriented information systems engineering. Unlike many other agent-oriented methodologies, RAP/AOR is not confined to the development of AI systems, but rather targets the development of large-scale distributed and cross-enterprise business information systems. Two particular strengths of RAP/AOR are its ontological foundation and its use of simulation for achieving more agility.

7. References

[1] Wagner, G. The Agent-Object-Relationship Meta-Model: Towards a Unified View of State and Behavior. *Information Systems 28:5 (2003)*, 475-504. See http://aor.rezearch.info/.

[2] Taveter, K. *A Multi-Perspective Methodology for Agent-Oriented Business Modelling and Simulation*. Ph.D. thesis, Tallinn University of Technology, 2004 (ISBN 9985-59-439-8).

[3] Kruchten, P. *Rational Unified Process – An Intoduction*. Addison-Wesley, 1999.

[4] Martin Fowler. *The New Methodology*. http://martinfowler.com/articles/newMethodology.html#N400315 [captured 29 June 2004].

[5] *OMG Model Driven Architecture*, http://www.omg.org/mda/

[6] Wagner, G., Guizzardi, G., Guarino, N., Sinderen, M. van. An Ontologically Well-Founded Profile for UML Conceptual Models. In: Persson, A., Stirna, J. (Eds.), *Advanced Information Systems Engineering, 16th International Conference CAiSE 2004, Riga, Latvia, June 7-11, 2004, Proceedings*. Lecture Notes in Computer Science (LNCS), Vol. 3084. Spinger-Verlag, 2004.

[7] Wagner, G. A UML Profile for External AOR Models. In: Giunchiglia, F., Odell, J., Weiss, G. (Eds.), *Agent-Oriented Software Engineering III, Third International Workshop, AOSE 2002, Bologna, Italy, July 15, 2002, Revised Papers and Invited Contributions*. Lecture Notes in Computer Science, Vol. 2585. Springer-Verlag, 2003.

[8] Sowa, J. F., Zachman, J. A. Extending and formalizing the framework for information systems architecture*. IBM Systems Journal 31 (3) (1992)*.

[9] *RosettaNet*, http://www.rosettanet.org.

[10] *OMG Unified Modeling Language Specification*, version 1.5, March 2003, http://www.uml.org/ [captured 30 June 2004].

[11] Yu, E. *Modeling Strategic Relationships for Process Reengineering*. PhD thesis, Department of Computer Science, University of Toronto, 1995.

[12] *Workflow Patterns*, http://tmitwww.tm.tue.nl/research/patterns/ [captured 30 June 2004].

[13] Wagner, G., Tulba, F. Agent-Oriented Modeling and Agent-Based Simulation. In: Giorgini, P., Henderson-Sellers, B. (Eds.), *Conceptual Modeling for Novel Application Domains*. Lecture Notes in Computer Science, Vol. 2814. Springer-Verlag, 2003.